

❖ Image Control :-

- It enables us to display images on webform page.
- We can also handle the images using server side code.
- To display a static image, we use tag.
- We can set the image to be display using ImageUrl property.
- Properties of Image Control :
 1. AlternateText
 2. ImageAlign: Specifies the alignment of the image. Values are NotSet, BaseLine, Bottom, Left, Right, Middle, TextTop.
 3. ImageUrl: It Specifies the URL of the image to be displayed.
- E.g

```
<asp:Image      ID="img1"      AlternateText="ABC"      runat="server"
ImageUrl="images\abc.jpg" ImageAlign="Left">
</asp:Image>
```

❖ List Controls :-

- There are 5 list controls in ASP.NET tool box. They are –
 1. DropDownList
 2. ListBox
 3. CheckBoxList
 4. RadioButtonList
 5. BulletedList
- All these controls are derived form System.WebControls.ListControls
- All the 5 controls differ in the way they render themselves and their associated ListItem.
- They all represent instance of ListItem class.
- They also contain Items collection with members corresponding to individual items in a list.
- We can add or remove items at compile time or run time.

1. DropDownList :-

- Enables user to select a single selection from dropdownlist.
- It is similar to listbox server control.
- The only difference is it shows only the selected item and other items are displayed when user clicks on dropdown button.

- It also supports dynamic databinding. (We can populate the ListItems at runtime using records from the table or database.)
- Properties of DropDownList Control :-
 1. Item : It specifies the collection of individual item in the list.
 2. DataTextField: Name of database field to supply the text of items.
 3. DataValueField: Name of datasource to supply value of items.
 4. DataSource: DataSource that populates the items in the dropdown.
 5. AutoPostBack
 6. SelectedIndex: It gets or sets index of the selected items.
 7. SelectedValue: It gets or sets index of the selected value.
 8. SelectedItem: It gets or sets the text of selected item.
 9. Text: Displays the selected text.
- Events :
 1. SelectedIndexChanged
- E.g DropDownListEg.aspx :


```
<asp:DropDownList runat="server" ID="cmbstate" AutoPostBack="true" >
    <asp:ListItem Selected="true"> Gujarat </asp:ListItem>
    <asp:ListItem> Maharastra </asp:ListItem>
</asp:DropDownList>
<asp:DropDownList runat="server" ID="cmbcity"> </asp:DropDownList>
```
- DropDownListEg.aspx.vb :


```
Partial Class DropDownListEg
    Inherits System.Web.UI.Page
    Protected Sub cmbstate_SelectedIndexChanged (ByVal sender As object,
    ByVal e As System.EventArgs) Handles cmbstate.SelectedIndexChanged
        If cmbstate.SelectedItem.Text = "GUJARAT" then
            Cmbcity.Items.Add ("SURAT")
            Cmbcity.Items.Add ("VAPI")
        Else If cmbstate.SelectedItem.Text = "MAHARASTRA" then
            Cmbcity.Items.Add ("MUMBAI")
        End If
    End Sub
End Class
```

2. ListBox :

- It is same as the dropdownlist but we can see multiple items at the same time.
- We can also create a multiselection listbox control.
- To use a listbox, we specify the rows property to get the number of items visible in the list.
- To enable multiselection, we use SelectionMode property and set it to multiple.
- Properties of ListBox control :-
 1. Item
 2. DataTextFiled
 3. DataValueField
 4. DataSource
 5. AutoPostBack
 6. SelectedIndex
 7. SelectedValue
 8. SelectedItem
 9. Text
 10. Rows: Specifies number of item visible.
 11. SelectionMode: Values are single or multiple.
- Event :
 1. SelectedIndexChanged ()
- E.g : ListBoxEg.aspx:


```
<asp:ListBox runat="server" SelectionMode="multiple" ID="LMenu"
AutoPostBack="true"></asp:ListBox>
<asp:ListBox runat="server" SelectionMode="multiple" ID="LPrice"
></asp:ListBox>
<asp:TextBox ID="txtTotal" runat="Server"></asp:TextBox>
<asp:Button ID="btnTotal" Text="Total" runat="server"></asp:Button>
```
- ListBoxEg.aspx.vb :


```
Partial Class ListBoxEg
    Inherits System.Web.UI.Page
    Protected Sub Page_Load() Handles Me.Load
        If NOT ISPostBack then
            LMenu.Items.Add ("DHOSA")
            LMenu.Items.Add ("IDLI")
            LPrice.Items.Add ("70")
```

```

        LPrice.Items.Add ("35")
    End If
End Sub
Protected Sub LMenu_SelectedIndexChanged (ByVal sender As object,
ByVal e As System.EventArgs) Handles LMenu.SelectedIndexChanged
    Dim I As Integer
    FOR I=0 to LMenu.Items.Count - 1
        If LMenu.Items (i).Selected = True Then
            LPrice.Items (i).Selected = True
        End If
    NEXT
End Sub
Protected Sub btnTotal_Click() Handles btnTotal.Click
    Dim I As Integer
    Dim total As Integer
    Total = 0
    FOR I = 0 to LPrice.Items.Count - 1
        If LPrice.Items (i).Selected = true Then
            Total= total + Int (LPrice.Items (i).Text)
        End If
    NEXT
End Sub
End Class

```

3. CheckBoxList Control :

- It is used to provide a group of checkboxes for multiselection of item.
- Here all the checkboxes are grouped in a single control.
- Every individual checkbox belongs to items collection.
- To determine which items (checkboxes) are selected we use the selected property.
- We can generate this control dynamically or bind to a data source.
- Properties of checkboxlist control :
 1. Items
 2. DataTextField
 3. DataValueField

4. DataSource
 5. AutoPostBack
 6. SelectedIndex
 7. SelectedValue
 8. SelectedItem
 9. Text
 10. Selected
 11. TextAlign : Specifies the alignment of text.
 12. RepeatLayout : Values are "Table" & "Flow" .
 13. RepeatDirection : Values are "Horizontal" & "Vertical".
 14. RepeatColumns : The number of column to be display.
- Event :
 1. SelectedIndexChanged ()

- E.g CheckBoxListEg.aspx :-

```

<asp:CheckBoxList runat="server" ID="chkLanguages"
RepeatLayout="Flow" RepeatDirection="Horizontal" RepeatColumn="3"
TextAlign="right" >
    <asp:ListItem>English</asp:ListItem>
    <asp:ListItem>Hindi</asp:ListItem>
    .
    .
    .
</asp:CheckBoxList>
<asp:Button ID="btnSelect" runat="server" Text="Selected Languages"
></asp:Button>
<asp:ListBox ID="ListLang" runat="server"></asp:ListBox>

```

- CheckBoxList.aspx.vb :-

Partial Class CheckBoxEg

Inherits System.Web.UI.Page

Protected Sub btnSelected_Click (ByVal sender As object, ByVal e As System.EventArgs)

Dim i As Integer

FOR i = 0 to chkLanguages.Items.Count - 1

If chkLanguages.Items (i).Selected = true Then

```

        ListLang.Items.Add (chkLangages.Items (i).Text)
    End If
NEXT
End Sub
End Class

```

4. RadioButtonList Control :-

- It is a single control that groups a collection of radio buttons such that only one of them can be selected at a time.
- It contains the items collection to access the individual items of the radio buttons in the list.
- Properties of RadioButtonControl :-
 1. Items
 2. DataTextField
 3. DataValueFiled
 4. DataSource
 5. AutoPostBack
 6. SelectedIndex
 7. SelectedValue
 8. SelectedItem
 9. Text
 10. Selected
 11. TextAlign
 12. RepeatLayout
 13. RepeatColumn
- E.g RadioButtonListEg.aspx

```

<asp:RadioButtonList runat="server" ID="rdyear" RepeateLayout="Table"
RepeatDirection="Vertical" TextAlign="left" AutoPostBack="True" >

```

```

    <asp:ListItem>FY</asp:ListItem>

```

```

    <asp:ListItem>SY</asp:ListItem>

```

```

    <asp:ListItem>TY</asp:ListItem>

```

```

</asp:RadioButtonList>

```

```

<asp:Image ID="imgyear" runat="server" Height="100px" Width="200px" />

```

- RadioButtonListEg.aspx.vb :-

```

Partial class RadioButtonListEg
    Inherits System.Web.UI.Page
    Protected Sub rdyr_SelectedIndexChanged()
        If rdyr.SelectedItem.Text = "FY" Then
            Imgyear.ImageUrl = "Fybca.jpg"
        Else If rdyr.SelectedItem.Text = "SY" Then
            Imgyear.ImageUrl = "Sybca.jpg"
        Else If rdyr.SelectedItem.Text = "TY" Then
            Imgyear.ImageUrl = "Tybca.jpg"
        End If
    End Sub
End Class

```

5. BulletedList Control :

- It creates a list in bulleted form.
- The bullets can be circular or rectangle or alphabets or number.
- Like other list control we can dynamically bind a data source or statically render a bulleted list.
- Properties BulletedList control :
 1. DataTextField
 2. DataValueField
 3. Item
 4. DisplayMode : It determines how to display the items of the list. Values are Text, Hyperlink and LinkButton.
 5. FirstBulletNumber : Sets the starting bullet number for the first item in the list.
 6. BulletStyle : Specifies the style of bullet. Values are NotSet, Disc, Square, circle, Numbered, LowerAlpha, UpperAlpha, LowerRoman, CustomImage.
 7. BulletImageUrl
- E.g BulletListEg.aspx :-


```

<asp:BulletedList ID="blsub" runat="server" BulletStyle="Numbered"
DisplayMode="Hyperlink">
    <asp:ListItem>101-CS </asp:ListItem>
    <asp:ListItem>102-Maths </asp:ListItem>
</aps:BulletedList>

```

- We can set URL on click of every listitem by specifying value attribute for every item.

mohamedsohel.co.in

❖ Calendar Control :

- It displays month calendar from user can select dates.
- It displays one month calendar and we can move to previous or next month.
- It is highly customizable in terms of functionality & appearance.
- Properties of Calendar Control:-
 1. FirstDayOfWeek: Set first day of the week.
 2. NextPrevFormat: Specifies the format of next and previous month navigation element. Values are CustomText, FullMonth & ShortMonth.
 3. SelectionMode: Specifies the date selection mode. Values are None, Day, DayWeek, DayWeekMonth.
 4. SelectedDate: To get the selected date.
 5. DayNameFormat: Specifies the format of the days of week. Values are full, short, shortest, FirstLetter, FirstTwoLetters.
- Events:
 1. SelectionChanged
 2. DayRender
- E.g CalEg.aspx :

```
<asp:Calendar ID="bD" runat="server" BackColor="white"
DayNameFormat="Short" NextPrevFormat="Short">
    <SelectedDayStyle BackColor="red" />
    <SelectorStyle BackColor="#cccccc" />
    <WeekEndDayStyle BackColor="#FFFFCC" />
</asp:Calendar>
```

❖ AdRotator Control :-

- It is used to display a sequence of ad images.
- It does not give any animated effects like flash or GIF.
- It simply presents ad images that when clicked navigate to a new location.
- It is usually used to show the advertisement banner on a webpage.
- The source of the advertisement is written in XML format in a file.
- It displays an image and also updates it and its URL each time a webpage is refreshed.
- Each time a page is loaded in the browser, the advertisement is randomly selected from predefined list.
- Properties of AdRotator Control:
 1. AdvertisementFile: Specifies path of XML file which contains the Ad information.
 2. KeywordFilter: Specifies a keyword to filter specific type of ads.
 3. Target: Specifies the name of the browser window or frame to display the content of the page linked to the AdRotator control.
- Event :
 1. AdCreated
- Note: This control can be bind to an XML file and any other datasource.
- Structure of XML file :- (Ad file)
 - AdRotator control uses a separate XML file to store an Ad information such as location of images, url, alternative text etc.
 - The file starts with <Advertisements> and ends with </Advertisements> tag.
 - Inside these tags, we have a pair of <Ad> & </Ad> for every Advertisement to be displayed.
 - E.g Ad.xml

```
<Advertisements>
  <Ad>
    <ImageUrl>images/abc.jpg</ImageUrl>
    <NavigateUrl>Gallery.aspx</NavigateUrl>
    <AlternateText>Gallery</AlternateText>
    <Keyword>Nature</Keyword>
    <Impressions>200</Impressions>
  </Ad>
</Advertisements>
```

- <Advertisements> tag :-
It specifies beginning and end of the Advertisement tag. If there are multiple tag then only the first tag will be parsed and remaining will be ignored by the compiler.
- <Ad> tag :-
It specifies a single image for advertisement. We can have multiple Ad tags for number of advertisement images.
It has the following properties as child tags.
 1. <ImageUrl> :-
It specifies the URL of image to be displayed.
 2. <NavigateUrl> :-
It specifies the URL of webpage to be redirected when the image is click.
 3. <AlternateText> :-
It specifies the alternative text when the images not displayed.
 4. <Keyword> :-
It specifies a keyword for filtering images form a set of images. We can specify the filtering in AdRotator control using the keyword filter property.
 5. <Impressions> :-
It specifies a numeric value that indicates the display frequency of the given image in the file. Higher the impression, the more number of times the image will be displayed relative to other images.
- E.g Ad.aspx :-

```
<asp:AdRotator ID="Ad1" runat="server" AdvertisementFile="Ad.xml"
KeywordFilter="Nature">
</asp:AdRotator>
```

❖ Container or Groping Control :-

- Container control is used to group other control inside it. We can add controls in a container control statically or dynamically at run time.
- There are 3 grouping control in asp.net :
 1. Pane Control
 2. Placeholder Control
 3. Multiview Control

1. Panel Control :

- It is used as a container for other control. We can have any control inside a panel like a radio button, textbox etc. Usually this control is use to generate controls by code dynamically and to display and hide a particular group of controls. It is rendered as a div element of html.
- Properties :-
 1. BackImageUrl :- Specifies background image for panel control.
 2. GroupingText :- Specifies the title of panel by creating a groupbox.
 3. HorizontalAlign :- It specifies the horizontal alignment of the container. Values are right, left, justify or notset.
 4. ScrollBar :- Specifies whether a scrollbar will appear or not.
 5. Wrap :- Specifies whether the contents of panel will be wrapped or not.

- E.g : panelEg.aspx :-

```
<asp:RadioButtonList ID="rdyr" AutoPostBack="true" runat="server">
    <asp:ListItem> Fybca </asp:ListItem>
    <asp:ListItem> Sybca </asp:ListItem>
    <asp:ListItem> Tybca </asp:ListItem>
</asp:RadioButtonList>
<asp:Panel ID="PanFy" runat="server" GroupingText="Fybca" Visible="false">
    FYBCA Subjects- C, Maths
</asp:Panel>
<asp:Panel ID="PanSy" runat="server" GroupingText="Sybca" Visible="false">
    SYBCA Subjects- Java, Vb.net
</asp:Panel>
<asp:Panel ID="PanTy" runat="server" GroupingText="Tybca" Visible="false">
    FYBCA Subjects- php, Asp.net
</asp:Panel>
```

- panelEg.aspx.vb :-

Partial Class panelEg

Inherits System.Web.UI.Page

Protected Sub rdyr_SelectedIndexChanged (ByVal sender as object, ByVal e
As System.EventArgs) Handles rdyr.SelectedIndexChanged

Dim i As Integer

i = rdyr.SelectedIndex

If i = 0 Then

```

        PanFy.Visible = "True"
        PanSy.Visible = "False"
        PanTy.Visible = "False"
    Else If i = 1 Then
        PanFy.Visible = "False"
        PanSy.Visible = "True"
        PanTy.Visible = "False"
    End If
End Sub
End Class

```

2. Placeholder Control :-

- It does not have any visible o/p but is used as a place holder when we add controls at runtime.
- It reserves a space for us to dynamically add or remove control.
- We have Add and Remove method to add and remove controls dynamically.
- E.g PlaceholderEg.aspx :-

```

<asp:RadioButtonList ID="rdcontrol" runat="server" AutoPostBack="True" >
    <asp:ListItem> TextBox </asp:ListItem>
    <asp:ListItem> Button </asp:ListItem>
</asp:RadioButtonList>
<asp:Placeholder ID="p1Control" runat="server">
</asp:Placeholder>

```

- PlaceholderEg.aspx.vb :-

```

Partial Class PlaceholderEg
    Inherits System.Web.UI.Page
    Protected Sub rdcontrol_SelectedIndexChnaged (ByVal sender As object,
        ByVal e As System.EventArgs)
        Dim btn AS New Button
        Dim i As Integer
        i = rdcontrol.SelectedIndex
        Dim txt As New TextBox
        If i =0 Then
            p1Control.Controls.Add (txt)

```

```

Else If i =1 Then
    p1Control.Controls.Add (btn)
End If

End Sub

End Class

```

3. Multiview Control :-

- Sometimes we decide to separate the process of filling form into several steps (pages).
- In such situation, the Multiview control act as a container for groups of view control.
- A Multiview control has a <asp:View> tag which can be having group of controls that we required.
- A Multiview is a simple way to declare multiple views and only show one at a time.
- We can navigate through the views using ActiveViewIndex property.
- To navigate through different views, we need to put button on the views with command name as next view and previous view.
- E.g MultiViewEg.aspx :-

```

<asp:MultiView ID="f1" runat="server" ActiveViewIndex="0">
    <asp:View ID="view1" runat="server">
        <asp:TextBox ID="txtname" runat="server">
        <asp:TextBox ID="txtage" runat="server">
        <asp:Button ID="cmdnext" runat="server" text=">" />
    </asp:View>
    <asp:View ID="view2" runat="server">
        <asp:TextBox ID="txtphno" runat="server">
        <asp:TextBox ID="txtemail" runat="server">
        <asp:Button ID="cmdprevious" runat="server" text="<">
        <asp:Button ID="cmdnext1" runat="server" text=">">
    </asp:View>
    <asp:View ID="view3" runat="server">
        <asp:Label ID="lblname" runat="server">
        <asp:Label ID="lblage" runat="server">

```

```

        <asp:Label ID="lblphno" runat="server">
        <asp:Label ID="lblemail" runat="server">
        <asp:Button ID="cmdprevious1" runat="server" text="<">
        <asp:Button ID="cmdfinish" runat="server" text="Finish">
    </asp:View>
</asp:MultiView>

```

- MultiViewEg.aspx.vb :-

Partial Class MultiViewEg

Inherits System.Web.UI.Page

Protected Sub cmdnext_Click (ByVal sender As object, ByVal e As System.EventArgs) Handles cmdnext.Click

f1.ActiveViewIndex = 1

End Sub

Protected Sub cmdfinish_Click (ByVal sender As object, ByVal e As System.EventArgs) Handles cmdfinish.Click

lblname.text = txtname.text

lblage.text = txtage.text

lblphno.text = txtphno.text

lblemail.text = txtemail.text

End Sub

Protected Sub cmdprevious_Click (ByVal sender As object, ByVal e As System.EventArgs) Handles cmdprevious.Click

f1.ActiveViewIndex = 0

End Sub

Protected Sub cmdnext1_Click (ByVal sender As object, ByVal e As System.EventArgs) Handles cmdnext1.Click

f1.ActiveViewIndex = 2

End Sub

Protected Sub cmdprevious1_Click (ByVal sender As object, ByVal e As System.EventArgs) Handles cmdprevious1.Click

f1.ActiveViewIndex = 1

End Sub

End Class

❖ Validation Control :-

- After developing a web application, the main task of every developer is to check for errors.
- For this purpose we have a series of ASP.NET validation controls.
- These controls are used to validate the data of any webserver input control.
- If any input control violates the condition for validation an error message is display.
- ASP.NET does client side and server side validation automatically by performing browser detection while generating a webpage.
- If the browser supports JavaScript, it renders its controls to JavaScript and validation occurs on client side, otherwise on server side.
- Validation controls are used to :-
 1. Implement presentation logic.
 2. Validate user input data.
 3. Validate data format.
- Common properties of Validation controls :-
 1. ControlToValidate: Specifies the unique id of input control to be validated. It is a required property.
 2. SetFocusOnError: Boolean property which specifies whether to set the focus on input control when the value is invalid.
 3. Text: Specifies text to display when the value is invalid.
 4. ErrorMessage: Specifies the message when the value is invalid. (It is used only when we use ValidationSummary control).
- Note:
 - One validation control will validate only one input control but multiple validation controls can be assigned to a single input control.
 - It is compulsory to set CausesValidation property to true to perform the validation.

1. RequiredFieldValidator :-

- It is a simple validation control that checks whether any input control is blank or not.
- Usually it is used to keep validation on all compulsory fields.
- The validation fails also if the value set by default does not change.
- Properties of RequiredFieldValidator Control :-
 1. Display:

- Specifies the appearance of behavior of the control.
- Values are :-
 - “static” -> Space is reserved for message.
 - “dynamic” -> Space is not reserved for message.
 - “none”

2. InitialValue:

- Specifies the initial values for input control. Default is Null.
- E.g RequiredFieldValidatorEg.aspx :-
Name:

```
<asp:TextBox ID="txtName" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="reqName" runat="server"
ControlToValidate="txtName" Text="Please enter name ">
</asp:RequiredFieldValidator>
<asp:Button ID="btnsubmit" runat="server" Text="submit"
CausesValidation="True" />
```

2. RangeValidator Control :-

- It provides the user to give specific range of values to the user.
- It allows the user to check the range of string, integer, double, date, currency.
- Properties of RangeValidator Control :-
 1. Type:
 - Specifies the datatypes of the values to be entered. Values can be string, integer, double, date, currency.
 2. MinimumValue:
 3. MaximumValue:
 - Specifies the minimum and maximum value for the control.
- E.g RangeValidatorEg.aspx :-
Age:


```
<asp:TextBox ID="txtage" runat="server" >
</asp:TextBox>
```

```

<asp:RangeValidator ID="ValAge" runat="server" InitialValue="25"
Text="Enter Age">= 18 & <= 75" Type="Integer" MaximumValue="75"
MinimumValue="18" ControlToValidate="txtage">
</asp:RangeValidator>
<asp:Button runat="server" ID="btnsubmit" Text="submit"
CausesValidation="true" />

```

3. RegularExpressionValidation Control :-

- The validation checks whether the value in input control matches with a given pattern defined by the regular expression.
- If the validated control is empty, no validation takes place.
- It allows validation for zipcode, URL, emails, telephone no., postal codes, etc.
- If we don't have the available expression, we can use and define our own customized expression.

- Properties of RegularExpressionValidation Control :-

1. ValidationExpression:

- Specifies the regular expression to validate the control.

- E.g RegularExpressionValidationEg.aspx :-

```

<asp:RegularExpressionValidation ID="valuser" runat="server" Text="Enter
proper username" ControlToValidate="txtuser"
ValidationExpression="/^[a-zA-Z_0-9]{8,}$/">
</asp:RegularExpressionValidation>

```

4. CompareValidator Control :-

- It is used to compare the value of one input control to another input control or to a fixed value.
- The two controls are compared with the type of comparison which is specified by the operator attribute.
- On comparison, if the expression evaluates to true, the validation result is valid.

- Properties of CompareValidator Control :-

1. Type:

- Specifies the datatype of the values to compare.

2. Operator:

- Specifies the operator to perform the comparison. Values can be Equal, NotEqual, LessThan, GreaterThan, GreaterThanEqual, LessThanEqual & DataTypeCheck.

3. ValueToCompare:

- Specifies the value to be compared with given input control.

4. ControlToCompare:

- Specifies the control whose value will be compared with given control.

- E.g CompareValidatorEg.aspx :-

```
<asp:TextBox runat="server" ID="txtpass" />
<asp:TextBox runat="server" ID="confirmpass" />
<asp:CompareValidator runat="server" ID="cmppwd"
ControlToCompare="confirmpass" operator="Equal"
ControlToValidate="txtpass" Text="enter value" >
</asp:CompareValidator>
```

5. CustomValidator Control :-

- It allows us to customize the validation process by writing a user define handler to validate the value enter.
- It performs user define validation both on client or server side.
- It is used when we don't have available predefine validation control.

- Properties:-

1. Value: Specifies the value to be validated.
2. IsValid: Checks whether the value is valid or not.

- E.g Custom.aspx :-

```
<asp:TextBox ID="txtEven" runat="server" >
</asp:TextBox>
<asp:CustomValidator ID="valEven" runat="server"
ControlToValidator="txtEven" Text="Enter Even No">
</asp:CustomValidator>
```

- Custom.aspx.vb :-

Partial Class Custom

Inherits System.Web.UI.Page

Protected Sub valEven_ServerValidate (ByVal Source As object, ByVal args As System.Web.UI.WebControls.ServerValidateEventArgs) Handles valEven.ServerValidate

If txtEven.TextMod 2 = 0 Then

args.IsValid = True

Else

args.IsValid = False

End If

End Sub

End Class

6. ValidationSummary Control :-

- It is a reporting tool that lists all errors of all validation controls in a list manner.
- We can consolidate errors of all controls and display in single validation summary.
- The ErrorMessage property and Text property are used to display the error message but ErrorMessage has higher precedence than Text property.
- Properties of ValidationSummary Control :-
 1. DisplayMode: Specifies the display for the list of errors. Values can be – “List” ,“BulletedList” and “SingleParagraph”.
 2. ShowMessageBox: Specifies whether to display the message box or not.
 3. ShowSummary: Specifies whether to show summary of errors or not.
 4. HeaderText: Specifies the header of summary text.

❖ ADO.NET :

- DataSource Controls :-

- We can use SqlDataSource control along with any DataBound control to retrieve data from a relation database and display, edit, sort data on a webpage with little or no code.
- To connect to a database, we must set the Connection-String property to a valid string value.
- The SqlDataSource can support any relational database that can be connected using ADO.NET providers such as SqlClient, OracleClient, OleDb, odbc.
- We can use SqlDataSource along with controls like –
 1. ListBox
 2. CheckBoxLayout
 3. RadioButtonList
 4. DropDownList
 5. BulletedList
 6. GridView
 7. Repeater
 8. DataList
 9. FormView
 10. DetailsView
- We can also use DataSource of other types :-
 1. AccessDataSource
 2. LinqDataSource
 3. XMLDataSource
 4. SiteMapDataSource
 5. ObjectDataSource
- The DataBound controls can be associated with the datasource using the DataSourceID property.

❖ CheckBoxList Control (DataBound) :-

- E.g CheckBoxList.aspx :-

```
<asp:CheckBoxList ID="chkSubject" runat="server"
RepeatDirection="Horizontal" RepeatColumns="3" RepeatLayout="Table">
</asp:CheckBoxList>
```

- CheckBoxList.aspx.vb :-

Partial Class CheckBoxList

Inherits System.Web.UI.Page

Protected Sub Page_Load (ByVal sender As object, ByVal e As
System.EventArgs) Handles Me.Load

chkSubjectFill()

End Sub

Dim cn As New SqlConnection ("ConnectionString")

Sub chkSubjectFill()

Dim str As String

Dim da As SqlDataAdapter

Dim ds As DataSet

Str = "select * from subject"

Da = New SqlDataAdapter (str,cn)

Ds = New Dataset

Da.Fill (ds)

chkSubject.DataSource = ds.Table (0)

chkSubject.DataValuefield = ds.Table (0).Columns (0)

chkSubject.DataBind()

End Sub

End Class

❖ DataBound Control / DataBase Control :-

❖ GridView Control :-

- It is an enhanced version of ASP.NET 1.X datagrid control.
- It is the only DataBound control which has built-in functionality for edit, delete, update, sort and paging.
- We can enable this functionality without writing a single line code.
- It provides more flexibility in displaying and working with data from the database than any other control.
- We have many properties to customize the look and field of GridView control.
- The only disadvantage is that it rendered as an html table and so it is not possible to display data in other formats.
- Because each column of GridView is render as <td> tag of html.
- Properties of GridView Control :-
 1. DataSourceId :- Specifies the id of the datasource to be used with this control.
 2. AllowPaging :- Specifies whether the control should allow paging or not.
 3. AllowSorting :- Specifies whether the control should allow sorting or not.
 4. SortDirection :- Specifies to sort in ascending or descending order.
 5. AutoGenerateEditButton
AutoGenerateDeleteButton
AutoGenerateSelectButton :-
Specifies whether given buttons are created automatically as separate columns.
 6. AutoGenerateColumns :- Specifies whether to generate columns automatically from the given datasource.
 7. Rows :- Get a collection of GridViewRow objects.
 - 8.PageIndex :- Gets or Sets the page index number for paging property.
 9. PageCount :- Specifies the number of pages required to display the records.
 10. PageSize :- Specifies the number of records to be displayed on a page.
 11. DataKeyNames :- Gets an array that contains the name of primary key field currently displayed in the GridView.
 12. HeaderRow
 13. FooterRow

- 14. ShowHeader
- 15. ShowFooter
- 16. HorizontalAlign
- 17. CaptionAlign :- Specifies alignment of caption.
- 18. Caption :- Used to give caption.
- 19. GridLines
- 20. CellSpacing
- 21. CellPadding

- Events :-

- 1. PageIndexChanged :-

- 2. PageIndexChanging :-

Both the events occur when the page link is clicked. They are fired before and after the GridView handle paging.

- 3. RowCancelingEdit :-

It is fired when the cancel button is clicked when the GridView is in edit mode.

- 4. RowEditing :-

Fires when an edit button of a row is clicked but before the edit operation.

- 5. RowUpdating :-

- 6. RowUpdated :-

It is fired when update button of GridView is clicked. Both the events are fired before and after the GridView handle the update operation.

- 7. RowDeleting :-

- 8. RowDeleted :-

Fires when the delete button of a GridView row is clicked.

- 9. RowCommand :-

It is fired when a button is clicked on any row of GridView.

- 10. RowCreated

- 11. RowDataBound :-

Fired when a row is bound to the data.

- 12. Sorting :-

- 13. Sorted :-

Both events are fired when the column header is clicked.

- Methods of GridView :-

- DataBinding expression creates binding between any property on aspx page.
- It includes the server control properties and the datasource when the databind method is called on a page.
- There are 3 ways to bind data with a databound control.

1. Eval ():-

- Eval is used for defining one way binding to a database control.
- It becomes read only and we can not change the data.
- E.g labels, literals, etc.
- It takes the name of data field and returns a string containing the value of that field from the current record.
- `<%Eval ("imgName") %>`

2. Bind () :-

- It is used for 2 way binding so that we can display the values as well as edit them.
- E.g textbox, password, dropdown etc.
- The changes in this value will be reflected to the database also.
- `<%# Bind ("imgName") %>`

3. DataBind () :-

- This method is used to associate the datasource of a databound control with the control and render the data in that control.

- GridView Template Types :-

1. TemplateField :-

- To display any control within databound control we use TemplateField.
- A TemplateField is used when a developer needs a complete control over the contents of any databound control.
- It is represented by `<asp:TemplateField> ...</asp:TemplateField>`.
- It allows the developer to define actual looks of a control instead of a default appearance.
- It is supported by all the major databound controls.

- It has the following TemplateFields :-

1. HeaderTemplate :-

- It contains the template to define the header of the control.
- It is represented by <HeaderTemplate>...</HeaderTemplate>.

2. ItemTemplate :-

- It defines the row that is rendered in the GridView.
- It is a compulsory template.
- It is represented by <ItemTemplate>...</ItemTemplate>.

3. AlternatingItemTemplate :-

- It is used for displaying the alternate template for an itemtemplate. (usually for appearance)
- It is represented by <AlternatingItemTemplate>...</AlternatingItemTemplate>.

4. SeparatorTemplate :-

- It is used to display the content between 2 rows. (Usually the separation of 2 rows using <hr> tag.)
- It is represented by <SeparatorTemplate>...</SeparatorTemplate>.

5. FooterTemplate :-

- It is used to render the content in the footer row.
- It is represented by <FooterTemplate>...</FooterTemplate>.

2. Bound field:-

- Allows you to specify which columns to display.

3. Hyperlink field:-

- Display data as hyperlink.

- E.g Database

Tblstate – StateId, State

Tblcity – CityId, City, StateId

edit delete	1	Valsad	1
edit delete	2	baroda	1
edit delete	3	pune	2
edit delete	4	mumbai	2
edit delete	6	udaipur	3
edit delete	7	jaipur	3
edit delete	8	surat	1
edit delete	9	ahmedabad	1
add		<input type="text"/>	gujarat ▼
1 2			

update cancel	1	Valsad	gujarat ▼
edit delete	2	baroda	1
edit delete	3	pune	2
edit delete	4	mumbai	2
edit delete	6	udaipur	3
edit delete	7	jaipur	3
edit delete	8	surat	1
edit delete	9	ahmedabad	1
add		<input type="text"/>	gujarat ▼
1 2			

Grid.aspx :-

```
<asp:GridView ID="CityGrid" AllowPaging="True" runat="server"
    PageSize="8" DataKeyNames="CityId" AutoGenerateColumns="false"
    ShowFooter="True">
```

```
<columns>
```

```
<asp:TemplateField>
```

```
<ItemTemplate>
```

```
<asp:LinkButton ID="btnEdit" CommandName="Edit"
    runat="server"> Edit </asp:LinkButton>
```

```
<asp:LinkButton ID="btnDelete"
    CommandName="Delete" runat="server">
    Delete</asp:LinkButton>
```

```
</ItemTemplate>
```

```
<EditItemTemplate>
```

```
<asp:LinkButton ID="btnUpdate"
    CommandName="Update" runat="server"> Update
</asp:LinkButton>
```

```
<asp:LinkButton ID="btnCancel"
    CommandName="Cancel" runat="server">
    Cancel</asp:LinkButton>
```

```
</EditItemTemplate>
```

```
<FooterTemplate>
```

```
<asp:LinkButton ID="btnAdd" CommandName="Add"
runat="server"> Add </asp:LinkButton>
```

```
</FooterTemplate>
```

```
</asp:TemplateField>
```

```
<asp:TemplateField>
```

```
<ItemTemplate>
```

```
<asp:Label ID="lblCityId" runat="server" Text='<%=Eval
("CityId")%>' ></asp:Label>
```

```
</ItemTemplate>
```

```
<EditItemTemplate>
```

```
<asp:Label ID="lblECityId" runat="server" Text='<%=Eval
("CityId")%>' ></asp:Label>
```

```
</EditItemTemplate>
```

```
</asp:TemplateField>
```

```
<asp:TemplateField>
```

```
<ItemTemplate>
```

```
<asp:Label ID="lblCity" runat="server" Text='<%=Eval
("City")%>' ></asp:Label>
```

```
</ItemTemplate>
```

```
<EditItemTemplate>
```

```
<asp:TextBox ID="txtcityup" runat="server"
Text='<%=Eval ("City")%>' ></asp:Label>
```

```
</EditItemTemplate>
```

```
<FooterTemplate>
```

```
<asp:TextBox ID="txtcity" runat="server"> </asp:Label>
```

```
</FooterTemplate>
```

```
</asp:TemplateField>
```

```
<asp:TemplateField>
```

```
<ItemTemplate>
```

```

        <asp:Label ID="lblState" runat="server" Text='<#Eval
        ("State")%>' ></asp:Label>
    </ItemTemplate>
    <EditItemTemplate>
        <asp:DropDownList ID="ddlState" runat="server"
        AppendDataBoundItems="true" >
    </asp: DropDownList >
        <asp:HiddenField runat="server" ID="hdnVal"
        Value="<#Eval ("StateId") %>" > </asp:HiddenField>

    </EditItemTemplate>

    <FooterTemplate>

        <asp:DropDownList ID="ddlState1" runat="server"
        AppendDataBoundItems="true" >
    </asp: DropDownList >

    </FooterTemplate>

</asp:TemplateField>

</columns>

</asp:GridView>

```

- Grid.aspx.vb :-
Imports System.Data
Imports System.Data.SqlClient

Partial Class Grid

Inherits System.Web.UI.Page

Dim cnn As New SqlConnection ("Cnstr")

Protected Sub Page_Load () Handles Me.Load

Cnn.open ()

If Page.IsPostBack = false Then

GridFill ()

End If

End Sub

Sub GridFill ()

```

Dim str As String
Str = "select * from Tblcity"
Dim cmd As New SqlCommand (str,cnn)
Dim ds As New Dataset
Dim da As New SqlDataAdapter (cmd)
da.Fill (ds)
CityGrid.DataSource = ds
CityGrid.DataBind ()
End Sub
Protected Sub CityGrid_PageIndexChanging ()
    CityGrid.PageIndex = e.NewPageIndex
    GridFill ()
End Sub
Protected Sub CityGrid_RowEditing ()
    CityGrid.EditIndex = e.NewEditIndex
    GridFill ()
End Sub
Protected Sub CityGrid_RowCancelingEdit ()
    CityGrid.EditIndex = -1
    GridFill ()
End Sub
Protected Sub CityGrid_RowDeleting ()
    If cnn.State = ConnectionState.Closed Then
        cnn.open ()
    End If
    Dim lbl As Integer
    lbl = e.Keys (0)
    Dim str As String
    str = "delete from Tblcity where CityId = " & lbl
    Dim cmd As New SqlCommand (str,cnn)
    cmd.executeNonQuery ()
End Sub
Protected Sub CityGrid_RowUpdating ()
    Dim lbl As New Label
    Dim txt As New TextBox
    Dim drop As New DropDownList

```

```

lbl = CityGrid.Rows (e.RowIndex).FindControl ("lblCityId")
txt = CityGrid.Rows (e.RowIndex).FindControl ("txtcityup")
drop = CityGrid.Rows (e.RowIndex).FindControl ("ddlState")
Dim str As String
str = "update Tblcity set City='" & txt.Text & "'", StateId=" &
drop.SelectedValue & " where CityId=" & lbl.Text
Dim cmd As New SqlCommand (str,cnn)
cmd.ExecuteNonQuery ()
CityGrid.editIndex = -1
GridFill ()

```

End Sub

Protected Sub CityGrid_RowDataBound ()

```

Dim lblCID As New Label
Dim str As String
Str = "select * from Tblstate"
Dim cmd As New SqlCommand (str,cnn)
Dim ds As New Dataset
Dim da As New SqlDataAdapter (cmd)
da.Fill (ds)
If (e.Row.RowType = DataControlRowType.DataRow) AND
(e.Row.RowState AND DataControlRowState.Normal) =
DataControlRowState.Normal Then
    lblCID=e.Row.FindControl ("lblCityId")
End If
If (e.Row.RowType = DataControlRowType.DataRow) AND
(e.Row.RowState = DataControlRowState.Edit)
    Dim hdn As Hidden Field
    hdn=e.Row.FindControl("hdnsid")
    Dim DropEd As DropDownList
    DropEd=e.Row.FindControl ("ddlstate")
    DropEd.DataSource=ds
    DropEd.DataTextField=ds.Tables (0).Columns
("State").ToString ()
    DropEd.DataValueField=ds.Tables (0).Columns
("StateId").ToString ()
    DropEd.DataBind ()

```

```

DropEd.Items.Insert (0,New ListItem ("---select---
",0))

DropEd.Items.FindValue
(hdnVal.Value).selected=true

End If
If (e.Row.RowType = DataControlRowType.Footer)
    Dim DropAdd As DropDownList
    DropAdd=e.Row.FindControl ("ddlState1")
    DropAdd.DataSource=ds
    DropAdd.DataTextField=ds.Tables (0).Columns
("State").ToString()
    DropAdd.DataValueField=ds.Tables (0).Columns
("StateId").ToString ()
    DropEd.DataBind ()
End If

End sub

Protected Sub CityGrid_RowCommand ()

    If e.CommandName="Add" Then
        Dim txt As New TextBox
        Dim drop As New DropDownList
        txt = CityGrid.FooterRow.FindControl ("txtcity")
        drop = CityGrid.FooterRow.FindControl ("ddlState1")
        Dim str As String
        str = "insert into tblcity (City,StateId) Values ('" &
txt.Text & "'," & drop.SelectedValue & ")"
        cmd = New SqlCommand (str,cnn)
        cmd.ExecuteNonQuery ()

    End Sub

End Class

```


❖ Repeater Control :-

- The repeater control is a basic container control used for customizing the display of one or more records from a datasource.
- It is completely driven by templates.
- It just repeats the HTML & ASP.NET controls that are placed inside a template block.
- It is very flexible & allows the user to display the list of records in user define layout.
- We have to explicitly declare all HTML layouts, formatting & style tags within the template block.
- It does not have a built-in user interface like other control (datalist, gridview etc).
- The following templates are available for repeater control.
 1. HeaderTemplate
 2. FooterTemplate
 3. ItemTemplate
 4. SeparatorTemplate
 5. AlternatingItemTemplate
- Note: We usually display records in a repeater without changes (records can not be edited).
- Events:
 1. DataBinding
 2. ItemCommand
 3. ItemCreated
 4. ItemDataBound
- Example : RepeaterEg.aspx :-

```
<asp:Repeater ID="CityR" runat="server">
  <ItemTemplate>
    <Table border="1">
      <tr>
        <td><%=Eval ("cityId") %></td>
      </tr>
      <tr>
        <td><%=Eval ("city") %></td>
      </tr>
      <tr>
        <td><%=Eval ("stateId") %></td>
      </tr>
    </Table>
```

</ItemTemplate>

</asp:Repeater>

- RepeaterEg.aspx.vb :-

Partial Class RepeaterEg

Inherits System.Web.UI.Page

Dim cn As New SqlConnection (" ")

Protected Sub Page_Load ()

If page.IsPostBack = false Then

CityFill()

End if

End Sub

Sub CityFill ()

Dim str As String

str = "select * from tblcity"

Dim cmd As New SqlCommand (str,cn)

Dim ds As New Dataset

Dim da As New SqlDataAdapter (cmd)

da.Fill (ds)

CityR.DataSource = ds

CityR.DataBind ()

End Sub

❖ DataList Control :-

- It is a flexible, full featured container control used for customizing the display of one or more records from a datasource.
- It is an extended version of repeater control.
- It is used to display dataitems in a repeating list & also allows selecting & editing of items.
- The content & layout of item in datalist is defined using the template.
- When the page runs, it loops through the datasource & renders the content for each record as a datalist item object.
- Like repeater it also has a user define layout.

- The main advantage of datalist control over other control is that it also supports directional rendering.
- It does not specify predefined layouts for displaying & updating data.
- Properties:-
 1. CellPadding
 2. CellSpacing
 3. RepeatDirection: Values are "Horizontal" and "Vertical". Specifies the direction of items to be render.
 4. RepeatColumns: Specifies the no. of column when the direction is horizontal.
 5. RepeatLayout: Specifies the layout of the items.
 6. ShowHeader
 7. ShowFooter
- Templates:
 1. ItemTemplate
 2. HeaderTemplate
 3. FooterTemplate
 4. EditItemTemplate
 5. AlternatingItemTemplate
 6. SeparatorTemplate
 7. SelectedItemTemplate
- E.g DataListEg.aspx :-

```

<asp:DataList ID="dLProd" runat="server" RepeatDirection="Horizontal"
RepeatLayout="Table">
    <ItemTemplate>
        <Table CellPadding="0" CellSpacing="0" Border="1">
            <tr>
                <td>
                    <asp:Image runat="server" ID="imgProd" Height="100"
                    Width="100" ImageUrl='<%=# "images/" + Eval ("Prod_img")
                    %>' />
                </td>
            </tr>
            <tr>
                <td>
                    <asp:Label runat="server" ID="lblprod_Name" text='<%=#Eval
                    ("Prod_Name") %>' />
                </td>
            </tr>
        </Table>
    </ItemTemplate>
</asp:DataList>

```

```

                </td>
            </tr>
            <tr>
                <td>
                    <asp:Label runat="server" ID="lblprice" text='<#Eval ("Price")
                    %>' >
                </td>

            </tr>

            <tr>
                <td>
                    <asp:LinkButton          runat="server"          ID="lnkview"
                    NavigateUrl='<# "ProductDetails.aspx?ID=" + Eval ("Prod_Id")
                    %>' >
                        ViewMore
                    </asp:LinkButton>
                </td>

            </tr>

        </Table>

    </ItemTemplate>

</asp:DataList>

```

❖ FormView Control :-

- It is databound control and is completely template driven.
- It allows us to customize the display of a single row of data and offers built-in support for editing the data in the control.
- We can configure a form view control to enable adding, editing or deleting data from it.
- It will automatically page the data if there are multiple records like datalist and repeater, it does not leave predefined layout.
- It is typically used in master detail scenario where the selected record of the master control determines the record to be displayed in FormView control.
- The main difference between DetailsView and FormView is that DetailsView has default tabular rendering and FormView has user defined rendering.

- Properties:
 1. AllowPaging
 2. AutoGenerateEditButton
 3. AutoGenerateDeleteButton
 4. CellSpacing
 5. CellPadding
 6. PageCount
 7. PageIndex
 8. Rows
 9. HeaderRow
 10. FooterRow
 11. DefaultMode: Values are ReadOnly, Insert & Edit.
- Templates:
 1. EditItemTemplate
 2. ItemTemplate
 3. InsertItemTemplate

❖ DetailView Control :-

- It is a databound control that renders a single record at a time.
- It also provides navigation, insert, update & delete option.
- It is generally render as table tag of html.
- It is used to work as a master detail scenario where the selected record of the master control determines the record to be displayed in DetailsView control.
- It has a default rendering and we do not have to worry about the layout.
- Properties:
 1. AllowPaging
 2. AutoGenerateEditButton
 3. AutoGenerateDeleteButton
 4. CellSpacing
 5. CellPadding
 6. Pagecount
 - 7.PageIndex
 8. Rows
 9. HeaderRow
 10. FooterRow
 11. DeafultMode: Values are Insert, Edit & ReadOnly.

❖ State Management in Asp.net :-

- We application are based on the stateless protocol (HTTP) which does not retain any information about the request of the user.
- In client server communication using HTTP, the page is created every time it is requested.
- This behavior of HTTP initiates a new instant of a page for the same time every time we request the page.
- When it is postedback and the page is destroyed after the roundtrip.
- State management is defined as managing the state of one or more users or controls such as textboxes etc. in a webform.
- Because ASP.NET applications are accessed over stateless protocol HTTP, developers are forced to implement state management techniques when developing application which provide customize control.
- There are 2 types of state management techniques.
 1. Client side state management
 2. Server side state management

1. Client side state management:

- Here the information for managing the state is stored on the local machine.
- We have the following client side state management technique in Asp.net.
 1. ViewState
 2. HiddenField
 3. QueryString
 4. Cookies

a) ViewState:

- It is a default method to preserve the page & Control values between roundtrips.
- ViewState can be used to store the state information for a single user.
- It is a built-in feature which allows storing page specific information.
- In Asp.net, We can enable the ViewState at page level or control level by specifying `EnableViewState=true`.
- By default it is true.
- To assign or fetch the value of ViewState of any control, we use the ViewState property.

- E.g ViewState ("name") = txtname.Text
Lbl1.Text = ViewState ("name")
- Asp.net framework saves the value of the web page & each control in ViewState before rendering the page.
- When the page is rendered, the field with _VIEWSTATE name is created in the form of a hidden field.
- ViewState mechanism has performance overhead because the ViewState data is encoded as binary base file which increases its overhead by 30%.
- Advantages:
 1. Simple to manage for page level data.
 2. It is encrypted.
- Disadvantages:
 1. Makes a page heavy because it consumes more memory.
 2. Device limitation.

b) HiddenField:

- It is used to store data at the page level but because they are hidden (Invisible), they are not rendered by the browser.
- A HiddenField stores a single value in its value attribute & it must be explicitly added to a page.
- When the page is posted to the server, HiddenField values are also sending along with other controls.
- Because Asp.net has ViewState for maintaining the state of control using a HiddenField is redundant.
- We must not use the HiddenField for storing any information i.e sensitive or that relies on the working of your application.
- E.g .aspx:-

```
<asp:HiddenField ID="hdnstore" runat="server">
</asp:HiddenField>
```
- .aspx.vb :-

```
Protected Sub btnstore_Click ()
    Hdnstore.value = txtA.Text
End Sub
Protected Sub btndisplay_Click ()
    txtB.Text = Hdnstore.Text
```


End Sub

- Advantages:
 1. No server resource is required.
 2. Supported by all devices & browser.
 3. Simple & easy to implement.
 4. Small in size because it stores only a single value.
- Disadvantages:
 1. Less secure because when information is passed from one page to another, the values can be intercepted.
 2. Storing large values can cost performance slow down.
 3. Storage limitation.

c) QueryString:

- It is a piece of information i.e appended at the end of a Url.
- They are usually used to send information from one page to another in the form of plain text in the Url.
- <http://www.abc.com/details.aspx?v1=val1&v2=val2...>
- In above Url the QueryString starts from the question mark (?).
- It includes the attributes & value pairs in the form of v1=val1&v2=val2 and so on.
- Here v1, v2, etc are QueryString variables & val1, val2, etc are the value of QueryString variable collected from the web server controls on a page.
- E.g Assume that we have a list of products on one page & we have a link “view/read more” which shows the details of the selected product on another page.
- Here we use QueryString to pass the prod_id of the selected product to the next page.
- E.g Product.aspx
`<asp:Hyperlink ID="hyplink" runat="server" NavigateUrl='< %#Eval ("prod_id","details.aspx>ID={0}")%>' >`
View More
`</asp:Hyperlink>`
- Suppose we have clicked the first product, the Url will be..
<http://www.abc.com/details.aspx?ID=1>

- On the next page details.aspx , we can retrieve the QueryString value using the following syntax-

Dim ID As String

ID = request.querystring ("ID")

- Advantages:-
 1. Simple & easy to implement
 2. No server resources are required.
 3. Supported by all devices & browsers.
- Disadvantages:-
 1. It is very less secure because everything is visible in address bar & hence can be change.
 2. There is a limitation on length of Url.
(Urls can pass only 1024 KBs of data).

d) Cookies:

- Cookies are a small amount of information i.e stored in plain text file on the client machine or in memory of the browser session.
- Cookies are handy when a particular uses needs a specific piece of data for a specific period of time.
- It can remain till the life of a browser window or as long as months or years specified.
- All the cookie information is stored in pure text form.
- The cookie file can contain username, password, email etc. and other user performances that a user enters on a web page while surfing a website.
- Suppose a website that displays headlines will allow the user to select which type of news he wants to see.
- This information is stored in cookies so that when the user visits the website next time, it will only show the news that the user wants by taking the information about the news preference from the cookie file of that user.
- Advantages:
 1. Cookies work transparently without the use being aware that some information is stored.Therefore it is not always a good choice for storing complex, large & private information.

- Disadvantages:
 1. It is not very secure because the information is stored on the local machine in pure text form.
 2. If we explicitly disable cookies then security is preserve but many critical applications will not work.
 3. A cookie can have a limitation on size (max size is 4KB & per website cookies can be around 20KB).
- To use cookies in Asp.net we import System.net namespace.
 - We use the HttpCookie object to create a cookie.
 - Request & Response objects are used to manipulate a cookie & return a reference to HttpCookie object.
 - Cookies can be either persistent or temporary.
- How does a cookie work?
 - When a client request a page from the server for the first time, the server identifies that the client has no cookies & generates a cookie for it.
 - Then the server sends a cookie to the client which is saved on the client machine.
 - The request after that will be send to the server along with the cookie.
- E.g:
- Protected Sub btnlogin_Click ()

```

Dim tempcookie As New HttpCookie
Tempcookie.Values ("UserName") = txtUser.Text
Tempcookie.Values ("life") = DateTime.Now ().ToString ()
Tempcookie.Expires = DateTime.Now ().AddDays (1)
response.cookies.Add (tempcookie)

```

End Sub

Protected Sub page_load ()

```

txtuser.text=request.cookies (tempcookie.values
("username").ToString ())

```

End Sub

mohamedsohel.co.in

2. Server Side State Management :

a) Session:

- Asp.net allows to save values by using session state which is an instance of HttpSessionState for every active web application session.
- It is a solution for a problem for the cookie to store large, complex, private objects securely.
- Asp.net allows the programmer to keep any type of object in session and it is hidden from the user unlike query string.
- SessionState is scoped to current browser session and it is created for particular user only for a particular time.
- During a session unique identity of a user is maintained internally.
- The session will end when there is a session timeout or user explicitly ends it out by logging out.
- A collection of user defined session variables is termed as session state which is processed during a session.
- These variables are accessed using the session collection.
- Session variables can be destroyed after a particular period of time irrespective of its current state.
- Syntax:
Session ("keyword") = Value
var = Session ("keyname")
- E.g
Protected Sub btnlogin_Click () Handles btnlogin.Click
 Dim str As String
 str = "select user_id from tbluser where user_name=" & txtuser.Text
 & " and password=" & txtpassword.Text & "
 Dim cmd As New SqlCommand (str,cn)
 Dim r As Integer
 r = CInt (cmd.ExecuteScalar ())
 If r > 0 Then
 Session ("user_name") = txtuser.Text
 Response.Redirect ("Welcome.aspx")
 End If
End Sub
- Advantages:

1. Simple to implement
2. Data Persistent: Session state is persistent across application domain even when the server restarts or it is maintained on a different server.
- Disadvantages:
 1. Performance Consideration: Session variables will stay in memory till they are removed or replaced. Therefore they can degrade server performance.
- Properties and methods:
 1. SessionTimeout: Specifies the life time a session in minutes. By default it is 20 minutes.
 2. Session.Remove ("keyname"): Removes a specific session variable.
 3. Session.RemoveAll (): It destroys all session variable.
 4. Session.Abandon (): Terminates the current session.

b) Application State :-

- Application object is used to store frequently used data which does not change from user to user.
- It is implemented using System.Web.HttpApplicationStateClass.
- It provides methods which can be access by all the users across the application i.e the application object will be shared across multiple users.
- The application state stores a single value and it is reused many times till the application life exist.
- The application state variables are set and initialized at the time of loading the application for the first time.
- They are declared in the Application_Start event in the global.asax file.
- E.g Application ("cnt") = Application ("cnt") + 1
- Advantages:
 1. Simple and easy to use.
 2. Can be accessed globally.
- Disadvantages:
 1. Occupies the server memory for the entire application. Hence creates overhead.

mohamedsohel.co.in

❖ Asp.net built-in objects :-

- Asp.net has some built-in classes for easier web-development.
- They are:
 1. System.Web.HttpSessionState (Session object)
 2. System.Web.ApplicationState (Application object)
 3. System.Web.HttpResponse (Response object)
 4. System.Web.HttpRequest (Request object)
- System.Web.HttpRequest :-
 - In single words a web request is a request send from client machine to the server asking for a specific webpage.
 - It is send by the URL in a web browser or when we clicked a hyperlink on a webpage.
 - A request is usually send by a web browser in a proper format that the server can understand using http protocol.
 - A request include following information with a URL:
 1. Details about the browser.
 2. Information like screen resolution, IP address etc. of the user.
 3. Data input by user.
 4. Cookies stored on local machine.
 - Asp.net provides a class called http request which is defined in the System.Web namespace.
 - It provides methods and properties which help us to use various information related to a web request.
 - An instance of this class is created by default on all pages by the name request.
 - Methods and properties:
 1. Request.FilePath: This property returns the currently executed file path.
 2. Request.ServerVariables: It is used to extract specific information about the client who is making the request.

E.g Dim str As String

For Each str In Request.ServerVariables.AllKeys

Response.Write ("Key:" & str & "Value:" &
Request.ServerVariables (str))

Next

3. Request.Cookies: It is a collection which is used to retrieve the cookies on local machine.

E.g var = Request.Cookies ("tmpUser").Values ("user").ToString ()

4. Request.QueryString (): It is used to retrieve the QueryString variables passed in a URL from one page to another or on the same page.

E.g var = Request.QueryString ("Id")

- System.Web.HttpResponse:

- Response is the message sent from a web server to the client when the client makes a request.
- For each request from a client, the server gives a response unless there is an error.
- It includes several information about the requested page including cookies & the actual content to be displayed to the user.
- In Asp.net, we have HttpResponse class defined in System.Web to handle all responses received from the server.
- We have an instance of this class created by default with the name response.

- Methods and properties:

1. Response.Write (): This method is used to write dynamic to a webpage.

2. Response.Cookie (): It is used to send cookies to the browser.

E.g If we select remember me option, we can use cookies to send it to the browser.

Syntax: Response.Cookies ("user").Value = txtuser.Text

3. Response.Redirect (): It is used to redirect a user from one page to another.

mohamedsohel.co.in

❖ Difference between Response.Redirect and Server.Transfer :-

- Both the methods are used to transfer a user from one page to another.
- But Response.Redirect () redirects a request to a new URL and specifies the new URL while Server.Transfer specifies the current request and terminates the execution of current page and starts the execution of new page using the URL.
- Response.Redirect () can be used for both aspx and html pages while Server.Transfer () can be used only for aspx.
- Response.Redirect can be used to redirect to an external website while Server.Transfer redirect only on the website on the same server.
- When we use Server.Transfer, the previous page exists in memory and in Response.Redirect, the previous page is removed from server memory and a new page is loaded.
- Response.Redirect sends a request to a browser, the browser then sends to the server and the server delivers a response. So it is roundtrip. In Server.Transfer request is directly sends to the server. Hence there is no additional roundtrip.
- Therefore Response.Redirect is slower and Server.Transfer is faster.

❖ User Profile :-

- We can make a website more users friendly while customizing, according to the preferences of the users visiting the website.
- To customize a website according to user preferences, we need to store and collect user specific data that can be accessible throughout the website.
- In Asp.net, this can be done by using user profile.
- A user profile is collection of various properties that specify the user information that is to be gathered when a user visits a website.
- E.g. we can store firstname, lastname etc. in user profile.
- A user profile can consist of properties of both simple datatype as well as complex datatypes like class and structures.
- The advantage of user profile is that the information stored is permanently stored on server and does not get lost even when a user leaves a website.
- It is defined in the Web.config file in the form of XML.
- We can create 2 types of profiles in Asp.net
 1. Anonymous profile
 2. Authenticated Profile

1. Anonymous profile:

- There would be a situation where no. of users would like to explore the website for free before registering. So we need to provide attractive user profile to such user to make them register users of one website. Such surf for free users are called anonymous user because they do not have a registered username and password.
- To create anonymous profile we have to enable the anonymous identification feature (<anonymousIdentification>) while defining the profile.
- It generates unique identification number for each anonymous user visiting the website. This number is stored in a cookie on the anonymous user's machine.
- We need to define the <anonymousIdentification> tag in Machine.config or Web.config file.
- We create a profile property for a user in Web.config file which is used to store the information about the user.
- E.g Defining profile :-
<System.Web>

```

        <anonymousIdentification Enabled="True"
Cookies="AutoDetect|UseCookies|UseDeviceProfile|UseUri"
CookieName="" CookiePath=""
CookieProtection="None|Validation|Encryption|All" CookieTimeOut=""
Domain="" />
        <profile>
            <properties>
                <add name="fname" allowAnonymous="True" />
                <add name="age" type="System.Int16"
allowAnonymous="True" />
            </properties>
        </profile>
    </System.Web>

```

- AddProfile.aspx.vb:-

```

Protected Sub btnAdd_Click () Handles btnAdd.Click
    Profile.fname = txtname.Text
    Profile.age = Int16.Parse (txtage.Text)
End Sub

Sub Page_Load ()
    If Page.IsPostBack = false Then
        txtname.Text = Profile.fname
        txtage.Text = Profile.age
    End If
End Sub

```

2. Authenticated Profile:-

- When surf-for-free become registered user, we need to display different data to them. These users are called authenticated users and their profiles are called authenticated profile.
- Authenticated profiles are displayed to user only when they log on the website using valid username and password.
- By default all profiles are authenticated profile. They are made anonymous profiles by adding allowAnonymous = "true".

❖ User Controls :-

- User controls are easy way to create separate reusable visual components like other server controls.
- It reduces maintenance cost and increases reusability by implementing encapsulation.
- We can create a user control like an aspx file.
- It has an extension .ascx.
- We can use a user control anywhere on aspx page.
- Each user control contains display functionality and we can set up each control to retrieve different information using a single user control.
- We can have multiple instances of a user control.
- User controls fully support the web forms framework and so all user controls can have properties and events just like built-in server controls.

- Creating and using a user control:
 1. creating a user control file by right clicking on application folder => add new item.
 - Select web user control file.
 2. This file has an extension .ascx. We define the entire user interface element and the layout in the file.
 3. We define the code for the controls in .ascx.vb file.
 4. We can use this created control on any aspx page by registering it using @register directive.
- This user control is now embedded in an aspx file rather a standalone component.
- We need to specify some additional information for using the control.
- We specify TagName, TagPrefix and SourceFile in @register directive.
- This directive tells that we want to use a user control on our aspx page.
- The process of using a user control on aspx page is called consuming a control.
- E.g. usercontrol.aspx:-


```
<%@Control %>
<asp:DropDownList ID="drpCountry" runat="server" AutoPostBack="True" >
</asp:DropDownList>
<asp:DropDownList ID="drpState" runat="server" AutoPostBack="True" >
</asp:DropDownList>
<asp:DropDownList ID="drpCity" runat="server">
</asp:DropDownList>
```
- Usercontrol.aspx:-


```
<%@Page %>
<%@Register TagName="combo" TagPrefix="asp" Src="usercontrol.ascx" />
<asp:combo ID="cmb1" runat="server">
</asp:combo>
```
- Now we can retrieve value of user control on the aspx page by using the following.


```
Dim drp1 As New DropDownList
drp1 = cmb1.FindControl ("drpCountry")
```

❖ Web Services:-

- Web services are a communication platform between two different or same platform applications that allows using their web methods.
- E.g. It means that we can create a web services application in asp.net that can be used by a java based application by using methods of asp.net.
- In simple term, a web service is a way for interactive communication with objects over the internet.
- In web services a set of related standards enables any two computers to communicate and exchange data over a network.
- It does not have any user interface and only provides specific services to its customers.
- It also provides high level of abstraction and hence the web service consumer and producer knows only about each other inputs, outputs & location.
- A web service can be accessed by any application regardless of the software and hardware platform because it complies with the common industry standards.

They are:

1. SOAP (Simple Object Access Protocol)
2. WSDL (Web Services Description Language)
3. XML (Extensible Markup Language)

1. SOAP:

- It specifies a format for the XML message using web application to request and receive a web service from a server.
- These messages are called SOAP message.
- A SOAP message is transmitted across the web application using the http protocol.
- A SOAP message envelope contains detailed information required to convert a datatype to and from a remote component.

2. WSDL:

- It allows consumers of web service to know about it before interacting with it.
- It describes all information required to use a web service with the help of XML.
- A WSDL document contains operation, bindings and end points to a related to web service.

- Steps to create a web service:

1. Right click on the application folder -> add new item -> select the web service file having extension .asmx.

2. Web service .vb file is created in app_code folder of the application.

3. Layout for web service is defined in .asmx file.

4. The code for web service is written in .vb file.

- w1.asmx:

```
<%#WebService Language="VB" Class="w1"
CodeBehind="~/app_code/w1.vb" %>
```

- w1.vb:

```
Public Class w1
    Inherits System.Web.Services.webservice
    <WebMethod ()>
        Public Function CalSI (ByVal pr As Double, ByVal rt As Double,
ByVal t As Double) As Double
            Return pr*rt*t / 100
        End Function
    End Class
```

- After creating this web service, click on the web application solder and click on web reference.

- In the web reference, find out the web service with given name (w1).

- Click on add reference by giving it a name. Here simple SI. Now we use the simple SI to create an object of w1.

- Usew1.aspx.vb:

```
Imports System.Data
Partial Class Usew1
    Inherits System.Web.UI.Page
    Protected Sub btnsubmit (ByVal sender As object, ByVal e As
System.EventArgs) Handles btnsubmit.Click
        Dim wr As New SimpleInterest.Click
        Dim SI As Double
```

```
SI = wr.CalSI (Cdbl (txtpr.Text), Cdbl (txtRate.Text), CInt  
(txtTime.Text))  
lblSI.Text = SI  
End Sub  
End Class
```

- Advantages of web service:
 1. It works on cross platforms i.e. it can be implemented on any platform.
 2. Web services are loosely coupled and hence their interfaces and methods can be extended.
 3. They do not carry any state information with them.
 4. They are not affected by firewall.
 5. Its modular structures allow multiple organizations to communicate with same web services.
- Disadvantages of web services:
 1. It is slow & hence it is not visible for high performance requirement.
 2. Increases traffic on the network.
 3. Lack of security standards for web services.
 4. Some vendors want to retain their intellectual rights to certain web service standards.
- Application of web services:
 1. Web portal
 2. Weather reporting
 3. Stalk quotes.

❖ Navigation Control:

- It is used to help user navigate through a website.
- In Asp.net, we can store the hierarchy key of menu in a file which is called site map file and is available as web.sitemap in asp.net.
- This file is stored in the root directory of our web application.
- It is an XML file and describes the logical structures of our web application.
- We can define the layout of all the pages in the application and how they are related to one another.

- We can add or remove the page links to the sitemap file and managed the navigation of website effectively.
- Rules are:
 1. We can have more than one sitemap file in a project.
 2. Only one <Sitemap> tag must be there in that file.
 3. The <Sitemap> tag can have multiple <SiteMapNode> elements.

- Example:

```
<? XML version="1.0">
  <Sitemap>
    <SiteMapNode Url="Home.aspx" Title="Home" />
    <SiteMapNode Url="About.aspx" Title="About" />
    <SiteMapNode Url="" Title="Products" >
      <SiteMapNode Url="ProductCategory.aspx"
        Title="Category" />
      <SiteMapNode Url="Products.aspx"
        Title="Product" />
    </SiteMapNode>
  </Sitemap>
```

- Asp.net has following three navigation control:

1. SiteMapPath
2. MenuControl
3. TreeView Control

1. SiteMapPath:

- It displays navigation path which has links representing the user's current page till the root of the website.
- It is only know as BreadCrumb navigation or EyeBrow navigation.
- It obtains the navigation data from the SiteMapFile which includes the URL, description and location of the current page in the navigation hierarchy.
- It helps the user to know his current location in relation to rest of the website.

- Properties:

1. PathSeparator

2. NodeStyle

3. CurrentNodeStyle

4. RootNodeStyle

5. ShowToolTips

6. PathSeparatorStyle

- Example:

```
<asp:SiteMapPath ID="site1" runat="server" >  
    <NodeStyle BackColor="#ccffcc" />  
    <PathSeparatorStyle ForeColor="Red" />  
</asp:SiteMapPath>
```

2. Menu Control:

- It is used to display a large collection of objects known as menu items to the end user.
- It can be displayed horizontally or vertically.
- It can be used as databound control and receive data from the items collection at runtime.
- It allows supports binding data with the datasource using the SiteMapFile. For this we have to use SiteMap DataSource and assigned it to the DataSourceID property of menu control.
- It contains two types of menus.
 1. Static menu: It is always present on the web page.
 2. Dynamic menu: Whenever a user opens a parent menu it appears.
- Using XML data source control:
 - We can use the XML data source control to bind the data with the menu control.
 - XML datasource will bind all the XML data form XML file to the menu item property.
- Properties:
 1. Orientation: To set a direction in which to bind the data with the menu control.

2. StaticPopoutImageUrl: Set the URL of the image displayed to indicate that has a sub menu.
3. DynamicPopoutImageTextFormatString: Sets the alternative text to be use for the image to indicate that it has a sub menu.

- Style Properties:

1. Static/Dynamic menu style: Control the property of all menu item properties like horizontal padding or vertical padding.
2. DynamicMenuItemStyle: Controls the style of individual menu item property.
3. StaticSelectedStyle: Sets the appearance of the menu item selected by the user in static menu.
4. DynamicHoverStyle: Sets the appearance of a dynamic menu where the mouse pointer is hovered over it.
5. Items: It contains the entire menu item to the control.
6. SelectedValue: Get the value of selected menu item.

3. TreeView Control:

- We can use TreeView control for logical control of data similar to windows explorer.
- We can display the XML documents as well as database records in a tree structure.
- We can display the XML document as well as database records in a tree structures.
- We can also nest a TreeView control nodes.
- The content for TreeView control can be added in following way...
 1. By adding programmatically at runtime:

We can add the nodes using treeviewNodes property and its add () method.
 2. By adding nodes from an external datasource:

By setting DataSourceID property of a treeview control to appropriate file such as XML or database sql file. We can load treeview control from it.
 3. Adding nodes using editor while designing:

It is easy use but not flexible because it can not be changed at run time.